CLAIMS

What is claimed is:

1. A method of providing a dispatch table that reflects an execution context of computer readable instructions, the method comprising:

determining an execution context for computer readable instructions; and

responding to a change in the execution context by using a dispatch table which reflects the change.

2. A method according to Claim 1 wherein the dispatch table includes one or more addresses, each address associated with a starting location for code associated with an instruction to be interpreted.

3. A method according to Claim 2 wherein using a dispatch table which reflects the change in execution context further includes rewriting one or more entries of the dispatch table so that the rewritten locations point to code capable of handing the change in execution context.

4. A method according to Claim 2 wherein using a dispatch table which reflects the change in execution context further includes selecting a predefined dispatch table having one or more addresses that point to code capable of handing the change in execution context.

5. A method according to Claim 4 wherein the predefined dispatch table is selected from a library of predefined dispatch tables, where each dispatch table in the library reflects a specific execution context.

6. A method according to Claim 1 wherein determining a change in execution context further includes checking one or more memory references.

7. A method according to Claim 1 wherein the execution context for the instructions corresponds to the execution context of a thread (or more generally, schedulable object) executing in a data processing system.

8. A method according to Claim 7 wherein the dispatch table reflecting the change in execution context is capable of managing one of the following execution states of the thread:

unreachable state;

state in which reference or assignment rules are not required;

state in which all assignment rules are enforced;

state in which only reference rules are enforced;

state in which reference and assignment rules, except those that refer to a depth of a scope stack, are enforced; or

state in which all reference and assignment rules are checked.

9. A method according to Claim 7 wherein determining the execution context further includes determining whether the thread is a real-time thread.

10. A method according to Claim 9 wherein if the thread is a real-time thread, the execution context of the thread is one of: real-time thread with no scope on a scope stack; real-time thread with at least one scope on a scope stack; no-heap real-time thread with no scope on a scope stack; or no-heap real-time thread with at least one scope on the stack.

11. A method according to Claim 10 wherein each execution context is associated with a corresponding dispatch table.

12. A method according to Claim 11 wherein determining a change in execution context further includes:

determining that the execution context of the thread has changed; and

responding to the change by causing the thread to be associated with a dispatch table reflecting the change.

13. A method according to Claim 7 wherein determining the execution context further includes:

checking one or more memory references; or

checking one or more memory assignments.

14. A method according to Claim 12 wherein the memory is at least one of the following: scope memory, heap memory or immortal memory.

15. A method according to Claim 12 wherein checking memory references further includes detecting a change in execution context when there are at least two references to scoped memory areas on a scope stack.

16. A method according to Claim 7 further includes storing the previous execution context of the thread as reflected in the dispatch table before the change.

17. A method according to Claim 15 further includes in response to the thread returning to the previous execution context, causing the thread to be associated with the stored dispatch table.

18. A method according to Claim 1 wherein using a dispatch table reflecting the change in execution context causes the instructions to be interpreted using memory reference checking rules or memory assignment checking rules.

19. A method according to Claim 18 further including throwing an exception if there is a violation of any of the reference or assignment checking rules.

20. A method according to Claim 1 wherein the dispatch table is used to implement the instructions into machine executable code.

21. A method according to Claim 1 wherein a bytecode interpreter causes the dispatch table to reflect the change in execution context.

22. A method according to Claim 1 wherein the dispatch table is any one of: a bytecode vector table or opcode vector table.

23. A software system comprising:

a handler responsive to a change in execution context for computer readable instructions; and

an interpreter, coupled to the handler, which causes the instructions to be associated with a dispatch table that reflects the change in execution context.

24. A system for providing a dispatch table that reflects an execution context of computer readable instructions comprising:

means for determining an execution context for computer readable instructions; and

means for responding to a change in the execution context by using a dispatch table which reflects the change.

25. A method of managing state transitions of threads executing in a data processing system, the method comprising:

determining a state transition of an active thread; and

responding to the state transition by coupling the thread to computer readable instructions being capable of handing the state transition.

26. A method according to Claim 25 wherein coupling the thread to computer readable instructions further includes patching address entries in a dispatch table.

27. A software system for managing state transitions of threads executing in a data processor, the system comprising:

an handler responsive to a state transition of an active thread; and

an interpreter, in communication with the event handler, which couples the thread to computer readable instructions being capable of handing the state transition.

28. A system for managing state transitions of active threads executing in a data processor, the system comprising:

means for determining a state transition of an active thread; and

means for responding to the state transition by coupling the thread to computer readable instructions being capable of handing the state transition.